AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions of claims in the application:

Listing of Claims:

1. (Currently Amended) A remote file system that promotes truth on a client, comprising:

one or more client computers that operatively communicate with an online remote location to work on one or more file objects;

a caching component that selectively caches the one or more file objects to a local cache located on a respective client computer, thereby making it available to the client when disconnected from remote location, wherein the caching component silently pushes file objects added by the client to the remote location and triggers a corresponding directory change notification request when physical share connection state has changed, to facilitate effectively enumerating any affected directory; and

a component that resolves conflicts between a client version of the one or more file objects and a remote location version of the one or more file objects such that the client version overrides the remote location version when viewed on the client, wherein the component that resolves conflicts is based at least in part upon user preferences;

wherein modifications by the client while disconnected from the remote location are stored to the client's memory and then automatically uploaded to the remote location when the client regains connection to the remote location; ~~and~~

wherein transitioning online to the remote location is initiated by the caching component which periodically scans offline paths and then initiates an online transition when a path becomes reachable, once the caching component detects a reachable path, it sends an I/O (input/output) control to a driver to initiate an online transition on the reachable path, all existing handles still remain offline until each handle is backpatched individually, once the connection is online, the driver starts to backpatch the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out; and

a synchronization component that background synchronizes namespaces not in conflict between the client and the remote location; and

wherein the caching component flushes out stale data from local caches based upon comparison of file signatures and comparison of file properties, wherein the file properties comprise time stamp, file size, and revision count.

2. (Canceled)

3. (Previously Presented) The system of claim 1, wherein the component that resolves conflicts is based on a prioritization policy comprising an order of updating the file object.

4. (Previously Presented) The system of claim 1, wherein the caching component pushes modifications made to the file object back to the remote location to update the remote location version on a periodic basis.

5. (Previously Presented) The system of claim 4, wherein the periodic basis selected to maximize bandwidth usage and to mitigate potential data loss with respect to the client computer.

6. (Previously Presented) The system of claim 1, wherein the conflicts resulting from more than one client modifying the file object.

7. (Previously Presented) The system of claim 1, wherein the caching component writes a modified file object back to the remote location.

8. (Previously Presented) The system of claim 1, wherein the caching component caches modified data and writes back to the remote location at one of the following events:

at least before a corresponding handle closes; and

the remote location revokes write buffering.

9. (Previously Presented) The system of claim 1, wherein the caching component updates respective local caches when the file object is modified at the remote location while the respective client computers were disconnected from the remote location if local caches have not been updated while offline.

10. (Canceled)

11. (Canceled)

12. (Previously Presented) The system of claim 1, wherein the caching component trims the local cache based at least in part upon user preferences.

13. (Original) The system of claim 1, further comprising a viewing component that allows a merged directory view, the merged directory view comprising current files not in conflict, current files in conflict between the remote location and the client computer, and files that are newly generated on one of the client and remote location.

14. (Previously Presented) The system of claim 13, wherein the viewing component facilitates a merged directory view when the client and remote location become connected once again to visualize changes made to file objects on the client and on the remote location during an offline period.

15. (Previously Presented) The system of claim 13, wherein the newly generated files are not present on the client cache but are viewable by the client before the client cache is updated.

16. (Previously Presented) The system of claim 13, wherein the viewing component employs one or more visual or graphical enhancements to facilitate visualization of online conflicted files, offline conflicted files and overlays of files.

17. (Previously Presented) The system of claim 16, wherein overlays of files refers to overlaying client version of the file object over the remote location version of the file object to facilitate visualizing one or more changes made to the file object by at least one of the remote location and the client.

18. (Canceled)

19. (Canceled)

20. (Canceled)

21. (Original) The system of claim 1, wherein creation of a new directory on a client is always satisfied.

22. (Previously Presented) The system of claim 1, wherein the remote location comprises one or more servers.

23. (Currently Amended) A persistent caching method that facilitates truth on a client, comprising:

selectively caching one or more file objects from a remote server to at least one local cache located on at least one client computer while online;

transitioning to an offline state;

modifying by a client, a client-cached file object while offline;

viewing a client version of the file if it conflicts with the server version;

storing modifications by the client while offline in the client's memory;

automatically uploading the modifications to the remote location when the client is back online; ~~and~~

initiating an online transition *via* scanning offline paths until a path becomes reachable,

sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline until each handle is backpatched individually; and

backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out.

24. (Original) The method of claim 23, further comprising resolving conflicts between the client version and the remote server version based at least in part upon user preferences.

25. (Original) The system of claim 23, further comprising resolving conflicts based at least in part on a prioritization policy comprising a priority order of updating the file object.

26. (Original) The method of claim 23, further comprising pushing modifications made to the file object back to the remote server to update the remote server version on a periodic basis.

27. (Original) The method of claim 26, the periodic basis selected to maximize bandwidth usage and to mitigate potential data loss.

28. (Original) The method of claim 23, the conflicts resulting from more than one client modifying the same file object.

29. (Original) The method of claim 23, further comprising writing a modified file object back to or overwrite the local cache with the server version based on user preferences.

30. (Original) The method of claim 23, the caching component writes back to the remote server at one of the following events:

at least before a corresponding handle closes; and

the remote location revokes write buffering.

31. (Original) The method of claim 23, updating respective local caches when the file object is modified at the remote location while the respective client computers were disconnected from the remote location.

32. (Original) The method of claim 1, flushing out stale data from local caches based at least in part upon at least one of the following:

comparison of file signatures; and

comparison of file properties, the file properties comprising time stamp, file size, and revision count.

33. (Original) The method of claim 23, trimming the respective local caches based at least in part upon user preferences.

34. (Original) The method of claim 23, further comprising viewing a merged directory view of file objects, the merged directory view comprising current files not in conflict, current files in conflict between the remote location and the client computer, and files that are newly generated on one of the client and remote location.

35. (Original) The method of claim 34, further comprising viewing a merged directory view when the client and remote location become connected once again to visualize changes made to file objects on the client and on the remote location during an offline period.

36. (Original) The method of claim 34, the newly generated files are not present on the client cache but are viewable by the client before the client cache is updated and the newly generated files are not present on the server but are viewable by client before the server is updated.

37. (Original) The method of claim 34, employing one or more visual or graphical enhancements to facilitate visualization of online conflicted files, offline conflicted files and overlays of files.

38. (Original) The method of claim 16, overlays of files refers to overlaying client version of the file object over the remote location version of the file object to facilitate visualizing one or more changes made to the file object by at least one of the remote location and the client.

39. (Original) The method of claim 23, the caching component silently pushes file objects added by the client to the remote location.

40. (Original) The method of claim 23, further comprising performing background synchronizations of namespaces not in conflict between the client and the remote location.

41. (Original) The method of claim 23, the caching component triggers a corresponding directory change notification request whose physical share connection state has changed, to facilitate effectively enumerating any affected directory.

42. (Currently Amended) A persistent caching system ~~method~~ that facilitates truth on a client, the system stored on a computer readable storage medium and comprising a computer processor for executing the following software components:

means for selectively caching one or more file objects from a remote server to a means for at least one local cache located on at least one client computer while online;

means for transitioning to an offline state;

means for modifying a client-cached file object while offline;

means for viewing a client version of the file if it conflicts with the remote server version;

means for storing modifications by the client while offline, in the client's memory;

means for automatically uploading the modifications to the remote location when the client is back online; ~~and~~

means for initiating an online transition *via* scanning offline paths until a path becomes reachable;

means for sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline until each handle is backpatched individually; and

means for backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file

objects even when the file objects have been modified locally but the changes have not been pushed out.

43. (Canceled)

44. (Currently Amended) A computer readable storage medium comprising a computer processor for storing computer executable components of claim 1.